**ENTREGA 2 DEL PROYECTO**

# PREDICCIÓN DE LA SEVERIDAD DE UN ACCIDENTE DE TRÁNSITO EN FUNCIÓN DEL TIPO DE VEHÍCULO


**PRESENTADO POR:**

LIZETH ANDREA GIRALDO VELEZ
CC: 1001237603

ANGIE LISETH CORONEL YELA
CC: 1085333292

**PROFESOR:**

RAÚL RAMOS POLLÁN




**UNIVERSIDAD DE ANTIOQUIA**

**FACULTAD DE INGENIERÍA**


**INTRODUCCIÓN A LA INTELIGENCIA ARTIFICIAL**

1. **Exploración de los datos**

- Carga de los datos desde Kaggle: El dataset empleado contiene información sobre accidentes viales en Inglaterra desde 2005 hasta 2014, y toda la información se importa en el código empleando pocas líneas de código que permiten descargar los datos en el entorno virtual.

- Descripción del dataset: Se tienen 4 archivos, 3 de ellos son datasets que contienen más de 1 millón de filas con información detallada sobre el accidente, los vehículos implicados y las víctimas respectivamente, y el cuarto archivo describe cada una de las variables categóricas presentes en los 3 datasets.

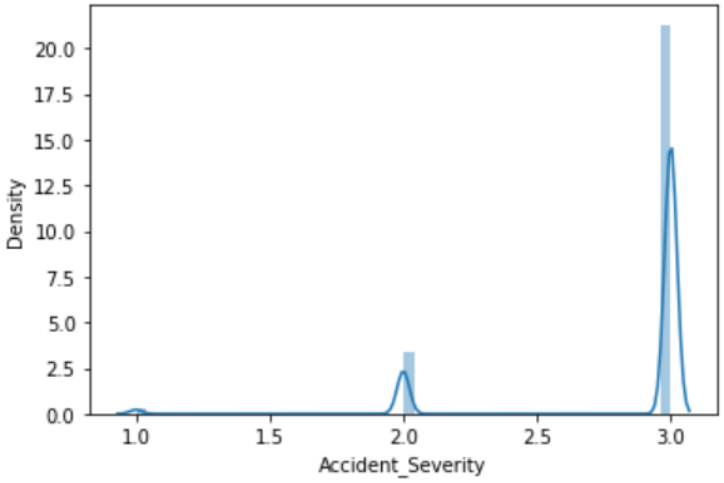
- Datos faltantes: En los 3 datasets hay datos faltantes, pero estos no están reportados como NaN sino como “-1”.

2. **Preprocesamiento de los datos**

- Identificación de las variables categóricas y numéricas: Una vez cargados los datasets, fue necesario identificar los tipos de variables para saber cómo llenar los valores faltantes, proceso que fue realizado teniendo en cuenta que, si son variables numéricas los datos faltantes se reemplazan por el promedio de todas las filas, si en cambio, son variables categóricas, son reemplazados por la moda.

3. **Modelos e iteraciones**

- Análisis de la variable objetivo `'Accident_Severity':` Se grafican los datos de la variable objetivo para conocer su distribución, evidenciándose que la mayoría de accidentes reportados son de tipo 3 o leves, seguidos de los accidentes tipo 2 o serios, y por último se encuentran los accidentes tipo 1 que son fatales.




- Generación tabla de variables finales de entrada y salida : Como se mencionó anteriormente, se tiene una gran cantidad de información reportada de los accidentes, por lo que se hace necesario eliminar variables que no están directamente relacionadas con el vehículo implicado en el accidente y conservar aquellas más relevantes que nos ayuden a entrenar el modelo.

**Variables de entrada:**

*['Road_Type', 'Speed_limit', 'Junction_Detail',*
*'Weather_Conditions', 'Road_Surface_Conditions',*

*'Special_Conditions_at_Site', 'Carriageway_Hazards',*
*'Urban_or_Rural_Area', 'Vehicle_Reference_x', 'Casualty_Severity',*
*'Vehicle_Reference_y', 'Vehicle_Type', 'Towing_and_Articulation',*
*'Vehicle_Manoeuvre', 'Junction_Location', 'Skidding_and_Overturning',*
*'Hit_Object_in_Carriageway', 'Hit_Object_off_Carriageway',*
*'1st_Point_of_Impact', 'Engine_Capacity_(CC)', 'Age_of_Vehicle']*

**Variable de salida:**

*['Accident_Severity']*

- División del Dataset en Train y Test

- Se realizó la separación del dataset concatenado, donde el 70% de los datos se usan para el entrenamiento del modelo, y el 30% para la validación.

```
[ ]  y = df_model_variables['Accident_Severity'].values
     X= df_model_variables.drop('Accident_Severity',axis=1)

     test_size = 0.3
     X_train, X_test, y_train , y_test = train_test_split(X, y, test_size=test_size,random_state=1)
     print(X_train.shape, X_test.shape)
```

```
(1148417, 21) (492180, 21)
```

- Aplicación de métodos supervisados

- Se probaron los siguientes modelos:

```
[ ]  estimator1 = LinearRegression()
     estimator2 = DecisionTreeRegressor(max_depth=5)
     estimator3 = RandomForestRegressor(n_estimators = 2,max_depth = 5)
```

- Para la elección del mejor estimador se obtuvieron los siguientes resultados

```
[ ]  test_size = 0.3
     val_size  = test_size/(1-test_size)
     zscores = []
     estimators = [estimator1, estimator2, estimator3]
     for estimator in estimators:
         print("---------")
         z = cross_validate(estimator, X_train, y_train, return_train_score=True, return_estimator=False,
                            scoring="neg_mean_squared_error", cv=ShuffleSplit(n_splits=10, test_size= val_size))
         report_cv_score(z)
         zscores.append(np.mean(np.sqrt(z['test_score']*(-1))))
     best = np.argmin(zscores)
     print ("Seleccionado: ", best)
     best_estimator = estimators[best]
     print ("\n Mejor modelo: ")
     print (best_estimator)
```

- Se prueba el mejor modelo elegido DecisionTreeRegressor

```
[ ]  print("Mejor estimador Decision Tree: ",decission_tree.best_estimator_)
     print("Mejores parámetros para el estimador Decision Tree: ", decission_tree.best_params_)
```

```
Mejor estimador Decision Tree:  DecisionTreeRegressor(max_depth=8)
Mejores parámetros para el estimador Decision Tree:  {'max_depth': 8}
```

- Función para calcular el RMSLE (Error logarítmico cuadrático medio) de los modelos implementados

```
RMSLE del Decision Tree en entrenamiento:  0.14001
RMSLE del Decision Tree seleccionado:  0.14037
```